AD-A015 322

DISTRIBUTED COMPUTATION AND TENEX-RELATED ACTIVITIES

J. Burchfield, et al

Bolt Beranek and Newman, Incorporated

Prepared for:

Office of Naval Research
Defense Advanced Research Projects Agency

August 1975

BOLT BERANEK AND NEWMAN INC

CONSULTING · DEVELOPMENT · RESEARCH

BBN Report No. 3117 August 1975

283118

DISTRIBUTED COMPUTATION AND TENEX-RELATED ACTIVITIES

Quarterly Progress Report No. 3

1 May 1975 to 30 July 1975

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 2901. Contract No. N0014-75-C-0773.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER BBN # 3117	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DISTRIBUTED COMPUTATION AND TENEX-RELATED ACTIVITIES		5. TYPE OF REPORT & PERIOD COVERED Quarterly Progress Report 5/1/75 - 7/30/75
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) J.Burchfiel, R.Thomas, T.Myer, R.Tomlinson		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0773
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, Mass. 02138		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE August 1975
		13. NUMBER OF PAGES 40
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 2935

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

distributed computation	message processing
TIP access control	CINCPAC interactive test
distributed file system	RSEXEC
TENEX security	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes continuing refinement of our TENEX RSEXEC distributed file system which supports geography-independent computing on a number of ARPANET TENEX sites. It also describes our efforts to upgrade existing ARPANET message service to meet NAVY requirements for an interactive message processing test at CINCPAC.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

I. Introduction

During the last quarter, we achieved significant progress in our Distributed Computation Research, our TENEX activities, and our COTCO Development effort.

In the Distributed Computation area, we improved the support facilities provided to TIP users through TIPSER/RSEXEC. In particular, it is no longer necessary for a TIP user to go through TIP login to read network news, and the network news is made available through a convenient message handling program. Another significant RSEXEC improvement was the incorporation of file-caching facilities into RSEXEC to improve performance and minimize delays by minimizing the number of files transferred across the network. Finally, we have started implementation of a general algorithm which solves the problem of coordinating multiple simultaneous updates to a distributed database. The example application we have chosen to demonstrate is updating of the network user ID database, which is used for both TIP login and netmail mailbox location.

Our TENEX activities this quarter included completion of an initial dispatcher and encapsulator which support use of arbitrary TENEX programs as tools within the National Software Works. In addition, the responsiveness of TENEX to network terminals was greatly improved through a number of changes to the TENEX scheduler and the NCP. (Network Control Program) Finally, the security of TENEX was substantially enhanced when we removed the File Transfer Server from the TENEX security perimeter.

Our COTCO activities resulted in the release of a new message system in April, and intensive review of this system by ARPA, the NAVY, and selected users. This review and feedback has resulted in significant revision of the design in order to meet the newly-identified requirements for the COTCO interactive text.

II. DISTRIBUTED COMPUTATION

A. TIP Support System

Two steps were taken to make it easier for TIP users to read the network news via the TIPSER/RSEXEC system (see BBN Report 2976). The first step was to permit users to access the NETNEWS command without requiring that they first successfully complete TIP login.

The second step was to provide an improved facility for reading items in the net news file. This was accomplished by modifying the NETNEWS command to use a version of the MSG system*, specially modified by USC-ISI for reading network news, rather than the READMAIL system. The old NETNEWS simply printed all of the items in the news file in reverse chronological order (i.e., most recent items first). Use of MSG by NETNEWS gives users the ability to survey the contents of the news file by printing the "headlines" of the news items and then to selectively print those items they wish to read.

Integration of MSG into the TIPSER/RSEXEC environment was a straightforward matter. It required no more than the exchange of a few network mail messages between BBN and ISI to agree upon the capabilities MSG should support and the conventions for the interface between MSG and the TIPSER/RSEXEC. The ease with which

- - - - -

* MSG is a program developed at the USC Information Sciences Institute for reading network mail. The network news file is maintained in the format of a mail system message file enabling it to be read by mail reading programs.

the integration was accomplished is due largely to use of the fork group features* of TENEX within the TIPSER/RSEXEC. These features make it relatively easy to incorporate independently developed software modules into the system.

TIP users who choose to login may, of course, still do so. After logging in, they are granted access to a wider range of TIP support services such as host status information, the ability to locate and link to other users, and a network mailbox locator function.

B. Distributed File System

Implementation of the "file caching" mechanism mentioned in our last QPR (BBN Report 3089) has been completed. This mechanism is designed to improve the performance of programs executing within the RSEXEC distributed file system environment by reducing the amount of network file transfer activity necessary to support access to remote files.

When a program attempts to access a file which is stored on a remote system, a check is made to see if a locally cached copy of the file exists. If there is a cached copy, and if the file in question has not been modified since the copy was created, it is

- - - - -

* These features include proxy login, per fork connected directories, and per fork controlling terminals. See BBN Report 2822.

unnecessary to retrieve a copy of the file and the program can be given access to the local copy immediately. If there is no such copy of the file, a copy must first be retrieved from the remote site before the program can be given access to the file. The copy retrieved will be maintained locally, for the duration of the user's RSEXEC session, for subsequent accesses by other programs.

We plan to make measurements to compare the performance of cached vs uncached operation. We expect these measurements will corroborate our subjective observations of improved responsiveness.

A beneficial side effect of implementing caching was to provide proper synchronization of multiple read and write accesses to remote files. Prior to implementation of caching, whenever a remote file was accessed the file would be retrieved but no record of the access would be maintained. Consequently, the cooperating TENEX systems did not synchronize simultaneous access to the file by multiple processes in the same way that a single TENEX system synchronizes such access.* This situation was corrected by opening the file at the remote site when a local program accesses it and keeping the remote file open for the duration of the access. As a result, attempts to access the remote file by other processes (local to or remote from the file) are subject to the normal single site TENEX

- - - - -

* Within a single TENEX, a file normally can have either a single writer or multiple readers. Simultaneous multiple writers, or simultaneous readers and writers, is not permitted unless all processes access the file in a special "thawed" mode.

synchronization constraints.

An issue that arises in supporting this sort of synchronization for remote file access is what to do if a process accessing a remote file crashes (e.g., its host crashes or the network partitions in a way that makes it impossible for the host systems involved to communicate). If no recovery procedure is defined for such situations, files maybe left in an open state indefinitely thereby preventing legitimate access by other processes. The recovery procedure used in the current implementation is to close the remote file if the communication path between the accessing process and the remote file is broken.

Another improvement to the distributed file system program execution environment was accomplished by implementing a new JSYS for TENEX. This new JSYS, called SETER, enables the RSEXEC process which intercepts file access operations made by executing processes (see BBN Report 3012) to properly return error codes when the file operations fail. The SETER JSYS will be distributed with version 1.34 of TENEX.

C. Maintenance of Distributed Data Bases

Within the context of the TIPSER/RSEXEC system, we have started an implementation of the technique reported in ARPA Network Working Group RFC #677 and BBN Report 2976 for maintaining distributed data bases. The purpose of this implementation is severalfold. It will give us an opportunity to evaluate the data base maintenance techniques we have developed by observing how they behave in practice.

The implementation will facilitate the maintenance of distributed copies of the network user ID data base used to support TIP login and the TIPSER/RSEXEC mailbox locator function. It will do this by allowing updates to the data base to be initiated from any of the data base sites and by guaranteeing that the updates are all incorporated into all data base copies in a consistent manner. In particular, users will be able to modify certain information the TIP maintains about them in the data base such as their passwords and network mailbox locations.

After this capability is made available to TIP users, it will be practical to make the mail sending functions of the network mail system available to TIP users again. These features were once available before TIP login was implemented, but they were removed because there was no way to reliably authenticate the identity of message senders. The TIP login process, together with the ability of a user to change his own password, makes it possible to reliably authenticate TIP users. We feel that the time is right, not only for re-installing the message sending function, but also for progressing further toward the realization of a highly reliable and conveniently accessible network mail service by making the message reading capabilities accessible through the TIPSER/RSEXEC system.

III. TENEX Related Activities

A. Internet Experiments

A simple gateway has been implemented which couples one logical network operating on the ARPA net using link 157 with another logical network operating on the ARPA net using link 158. The gateway is programmed in BCPL for the PDP-11. A parallel version of this program was implemented for TENEX in machine language. As of the end of this quarter, these gateways have not been extensively exercised.

In an attempt to improve the efficiency of the TCP operating under TENEX, the several processes of the TCP have been modified such that they run in one TENEX process. This reduces the total program activations needed to operate the TCP and should reduce time spent signalling and loading up each TCP process as it starts running. Comparative measurements of the two TCP versions have not yet been made.

That portion of the TENEX TCP code which does not interface with the operating system itself has been converted to run on the PDP-11. The new double precision features of the BCPL cross compiler has facilitated this task significantly.

B. The National Software Works

The past quarter of National Software Works (NSW) activity has been especially noteworthy due to a substantial change in the direction of the project as perceived from BBN. Since a major emphasis has been placed on a short term demonstratable system, we have been and are continuing to review earlier design decisions, many of which were formulated before BBN's active NSW participation. This has resulted in many of these initial decisions undergoing revision to reflect a change toward more limited goals and more immediate results.

We have considered our NSW involvement to run along three major concurrent paths. These are 1) defining the role and function of a Tool Bearing Host (TBH) in the NSW system 2) implementing these functions for the TENEX machine and 3) participating in the long range design of the NSW system, with special attention paid to identifying and proposing solutions to problems relating to the distributed nature of the system. Manpower restrictions and project goals have mandated that we concentrate our efforts on implementing the necessary set of TBH functions. We have recently completed an initial version of a network NSW Dispatcher program and a program to simulate the NSW environment to an arbitrary, running TENEX program. We have also spent a substantial amount of time conferring with the NSW group at Computer Associates on the design and implementation of an interim communication base for the NSW components.

1. NSW Network Request Dispatcher

The overall TBH design shows a concentrated effort toward insulating concurrent instances of tool use from adversely effecting each other. Thus our design specifies that there should exist a number of NSW working directories which are dynamically allocated on demand to NSW users for a tool session. This approach allows us to tap an existing file protection mechanism, already supported in most host systems. Complementary to this is the use of separate "jobs" for each NSW tool request. This takes advantage of a relatively new feature of the TENEX system which has proven very useful in implementing general network service programs. Using this technique, the NSW can rely on the same protection mechanisms that the host system provides to prevent local users from mutual interference.

A potential problem arose in the dispatcher design, owing to a drawback in the TENEX "create job" mechanism which makes it very difficult to find out about job termination. This information is needed so that network sockets dedicated to that job can be reclaimed and subsequently re-used by a new job. To use the dispatcher with only a finite socket space, we have programmed a technique whereby each new job creation causes the release of any sockets which were used by the previous incarnation of that job number. In this way, we are always assured of having sockets available for the next NSW request, provided we have an adequate initial supply of sockets.

The dispatcher was designed and coded using the concepts relating to the then current NSW network communication mechanism. It has been tested on a stand alone basis only. Changes in the underlying NSW communication base will probably require changes in the dispatcher module, but they are expected to be minor.

2. Establishing the NSW Environment on TENEX

The virtual machine provided for an NSW computer program (tool) is different from the TENEX virtual machine itself. The most prominent differences are in the handling of the file system and program error/termination conditions. We have selected encapsulation as the method most appropriate for creating the proper NSW environment on TENEX. This allows the easiest possible introduction of existing TENEX software as tools in the NSW. An initial design has been specified, and we have coded essential parts of this design for allowing an encapsulation program to intercept selected TENEX virtual machine system calls and instead invoke or implement them in an NSW sense. It is clear that this technique has limitations, mainly due to structures in either system which have no reasonable meaning or interpretation in the other. Despite this, we have successfully run a text editor and compiler in a limited stand alone test (since the ability to connect the relevant parts of the system is just now emerging).

The major effort in encapsulation has been to deal with the problem of interfacing file systems. Since an arbitrary program (tool) can reference files both internal and external to the NSW file system, there are two facets of the access control problem. The Works Manager retains responsibility for the majority of NSW access control checks. The TBH encapsulator handles local file access control through the use of access control lists obtained dynamically from the WM on a per-tool basis. The techniques for identifying the different classes of files are of an ad-hoc nature, as are many of the system call translation routines. Some of these will be formalized as we gain experience with the initial system and fully specify functionality and syntax.

There are two other functions of encapsulation which deserve mentioning. They were designed to reduce the number of network interactions necessary in performing routine operations in an attempt to increase performance. These functions are 1) local maintenance and manipulation of a crash proof dictionary relating NSW file names to their actual host system names and 2) use of the standard ARPANET TELNET communication mechanism to drive existing tools in a convenient, non-tool dependent manner.

3. Communication Support

There has been much controversy surrounding the communication support for the linkage of the NSW components. Late in the quarter, it was decided that in order to expedite the interconnection of NSW components, an interim support system would be upgraded to fill the immediate needs of component testing and concept demonstration. Toward this end, we have worked closely with the Computer Associates NSW group on initial efforts to achieve usable intra-host communication support as quickly as possible. Such an interim system should allow a partial system configuration early in the next quarter.

C. FTPSRV using CRJOB

As reported in our previous QPR, the Create Job JSYS (CRJOB) has been implemented in the TENEX operating system. One of the strongest motivations for that function was to provide a more secure and better accounted version of FTPSRV, the File Transfer Protocol Server program. This new version has now been implemented and is running on the BBN TENEX service systems.

The FTP server had been organized as one large job, with a process at the top and additional processes for each user of the FTP. The job ran as a privileged process, thus becoming part of the TENEX security kernel. It simulated the file access control of TENEX, and checked the users´ passwords itself. This led to a (fortunately small) number of errors in security. One was the result of a change in TENEX´s access control which was not copied into FTPSRV´s simulation of access control, resulting in the older, freer access being allowed through FTP when it was not allowed to an ordinary user. Another error was in the simulation of LOGIN, resulting in an unlikely erroneous typing sequence causing the checking of passwords to be bypassed.

The new organization of FTPSRV, using CRJOB, shifts all access checking back into TENEX where it belongs. TENEX´s normal LOGIN checks are used in response to the user´s name and password. Once logged in, the user´s job has only the privileges actually assigned to him. Therefore, TENEX's file access checks are applied and no extra capabilities can be given by any errors in FTPSRV.

Privileged operations are required to receive connections from the network via the assigned FTP socket number three. A very small program, only 420 words, does this task and is privileged. It performs the initial connection protocol and the CRJOB JSYS to connect the requesting user to a new job which will run the main body of FTP code. This small privileged control job (FTSCTL) takes no more action on behalf of the user.

Once connected to a new job, the user starts to run in a program called FTPSER. This server is a reorganization of the mail receiver and file operations portion of FTPSRV. The crucial part of this reorganization is that all operations which perform file access for the user are separated into a portion of the address space which is made inaccessible until a successful login operation is performed. The login itself is done through TENEX's LOGIN JSYS, thus assuring that a valid name, password, and account have been collected for the requesting user. The code is verifiably organized so that until a login is done, no file access commands can be performed.

One concession was made in favor of the continued existence of the "ANONYMOUS" use of FTP. This feature allows users to read or write only files which have completely public access for those operations. This feature was implemented in the earlier FTPSRV so that it would not be necessary for users to have an account on a

TENEX system in order to read public files from that system. This operation was controlled by a switch in the FTP server so that only those sites which wished to supply anonymous service would enable it. In the new FTPSER, anonymous use has been (optionally) retained through the creation of a real account named ANONYMOUS, which has no privileges. The password for ANONYMOUS is supplied to FTPSER through a read-protected system file, so that the remote user need not know it. When the remote user declares that his name is "ANONYMOUS", FTPSER uses the local password for ANONYMOUS to do the LOGIN JSYS.

This reduction in security was considered acceptable because:

(a) Only the password for ANONYMOUS is in a file. No others are.
(b) The file containing this password is read-protected.
(c) The TENEX EXEC was simultaneously modified to prevent logins by ANONYMOUS even if the right password is given.
(d) Any system not desiring to supply anonymous service can simply not create a user named ANONYMOUS, and the LOGIN JSYS will fail.

This FTP server re-implementation has resulted in a much safer system. No bugs are currently known in the old FTP server's access restrictions, but it is much better to have the new organization wherein no user-level privileged code need be trusted to protect the system's integrity.

D. TENEX Network Control Program Performance Improvements

1. High Priority Scheduling for Critical Sections

The Tenex network control program contains portions of code and certain tables which may be accessed by any process utilizing the network. To avoid chaos, these resources are protected by use of two network-related locks. Recently, complaints of slow echoing response when accessing Tenex via an NVT were traced to the following situation: a process with low scheduling priority would seize one of the network locks in order to access one of the above-mentioned resources, but due to its lack of priority would take inordinate amounts of time to complete the critical section and release the lock, thus making these resources unavailable to other processes. By way of curing this problem, a general mechanism was devised to provide special scheduling for processes executing critical sections of monitor code with special care given to insure that such processes cannot usurp the entire system by deliberately executing JSYSes, for example, which are known to contain critical sections.

2. Improved NVT Output Packing

During the process of moving characters from TTY output buffers to network buffers for output to the network, the characters pass through a small intermediate buffer, purely for implementation reasons. It was observed recently that the size of this buffer was having a profound influence on the size of NVT network output messages. This becomes serious when the source and destination host are many hops apart and RFNM delay is significant; it behooves one in such a situation to make output messages as large as possible to minimize the per-character RFNM overhead. The NCP has been modified accordingly such that NVT output message sizes are influenced only by number of characters in the TTY output buffer, or by allocation, or, ultimately, by the 8-packet limit imposed by the sub-net.

3. Fixed-size Input Buffers

It has been found that the network control program has a considerable appetite for processor cycles, particularly on heavily loaded network-only systems such as BBN. Using program counter sampling techniques, it was found that the NCP was spending fully 1/4 of its time assigning and releasing space for buffers. This was clearly due to the use of the same free storage package as is used for the Tenex file-system, a best-fit algorithm employing incremental garbage collection. While this is now known to be a sub-optimal approach to the storage management problem, it is clearly far less appropriate in the NCP buffer management context than in the file-system context where the calling frequency is lower. Analysis of the distribution of free storage requests made by the NCP led to the adoption of a fixed-size free storage management scheme. The NCP now consumes less than 2% of its time in buffer management at the sacrifice of some address-space but very little real memory, clearly a worthwhile improvement.

4. Fair Scheduling of Swap-Bound Processes

The 1.33 scheduler goes about the task of enforcing the pie-slice policy by dividing the schedulable processes into two categories: those 'behind-schedule' and those 'ahead-of-schedule', where this determination is made by comparing a process´ current cpu-utilization (actually an exponential average of same) with its current target utilization (its pie-slice group share divided by the number of processes currently active in the group). The behind-schedule processes are serviced in preference to the ahead-of-schedule processes, of course, but within each of these categories no distinction is made between processes, except to scale their quanta by a value proportional to their target utilizations. Thus, each category is serviced in something approximating a round-robin sequence. By failing to distinguish swap-limited from processor- limited tasks, highly concurrent operation of the processor and the swapper is not always achieved, thus inflating the average completion time of the tasks involved. Upon recognition of this deficiency, a new pie-slice regulator was implemented, which keeps a single queue of schedulable process, sorted by ´most-behind-schedule´ to ´least-behind-schedule´, thus making an N-way distinction amongst its processes, as compared to the 2-way distinction made previously. Since swap-bound tasks by definition accumulate cpu time less quickly than cpu-bound tasks, this regulator automatically achieves concurrent scheduling of the swapper and cpu.

5. Fast Process Wakeup Facility

In the past, while certain event handlers in Tenex were cognizant of the fact that a fork known to them would unblock as a result of the event they were processing, no mechanism existed for them to communicate their knowledge to the scheduler save for simply setting a flag which indicated to the scheduler that in fact SOME fork had awakened; it was left to the scheduler to then search for that fork -- a regrettable lack of communication. There now exists a routine in the scheduler for handling this situation; it is callable at process level and accepts a fork handle as an argument. It greatly reduces the overhead in awakening a fork in the circumstance described above. Examples of places in the monitor now using this feature are the PSI logic and JSYS traps.

E. BCPL

1. PDP-10 Compiler

This quarter, we have continued the coding of the new BCPL compiler for the PDP10. We discovered that the old code generators were poorly written, and needed a large amount of unexpected rewriting to make them work properly. All the code has been written, and we are nearly finished with the debugging.

The new compiler has successfully compiled several small programs and some medium-sized modules of larger programs, including the compiler itself. Results from the initial tests of the compiler indicate a saving of [illegible]m 20% to 25% in code generated (when loaded without local symbols). We have not been able to reliably test the difference in speed between code compiled by the old and new compilers, but expect that the difference will be quite significant. (Unfortunately, the new compiler is slower than the old one, since it does not have the ability to generate REL files directly -- it generates a MACRO file and then calls in MACRO to assemble the code. We hope to be able to implement direct REL file generation sometime in the next two quarters.)

We also resolved a question of compatibility between the old compiler, the new compiler, the library and the debugger (BDDT) by planning the release of the new BCPL system as follows:

(a) When the compiler has compiled itself with no (known) errors, the active bcpl users will notified of its existence. It will be placed on the <SUBSYS> directory on system A under a special name, and will be available through CCL by using a special extension. This version of the compiler will normally generate code compatible with the code currently generated by the old compiler. However, a compiler switch exists to make it generate "new format" calling sequence -- somewhat longer at execution time, but significantly shorter in code.

(b) After a satisfactory check-out from the active users, a new version of the BCPL library and BDDT will be written and placed on <SUBSYS>. The compiler will be changed to create as a default the new calling sequence, and will be renamed BCPL. A system message will inform users that the new compiler does NOT generate compatible code, and that all code must be recompiled.

2. PDP-11 Cross-Compiler

The considerable difficulty of dealing with double precision numbers for programming the TCP and other similar routines for the PDP-11 led to the addition of a double precision capability to the BCPL cross compiler for the PDP-11. This capability permits operations on 32-bit integers to be specified directly in the source code rather than through the awkward, inefficient, and error-prone use of subroutines. Double precision variables may be declared, initialized, and passed as routine arguments. No provision was made for returning double precision results either from functions or valof blocks.

Double precision numbers are stored in two adjacent words of memory. The low order word is stored in the word with the lower address and the double precision variable is addressed by that word's address. The programmer is insulated from details of variable storage by certain conventions described below.

Space for double precision variables is allocated by preceding the initial value of the variable with the double operator. Routine parameters are treated as double precision when preceded by the double operator. The double operator is automatically supplied when reference is made to a double word field of a structure. As usual with BCPL, the programmer is responsible for matching data types between function call and definition and between operands and operators.

There are three double precision binary arithmetic operators, 10 double precision relational operators, and 4 double precision unary operators. The double precision arithmetic operators perform addition, subtraction, and multiplication of two double precision operands producing a double precision result (%+, %-, %*). The relational operators perform the usual set of comparisons including unsigned magnitudes (dgr, dge, dle, dls, deq, dne, dgrm, dgem, dlem, dlsm). The unary operators are double (treat following expression as double precision), %- (negation), extend (extend single precision signed integer), and extendm (extend single precision unsigned integer).

The version of the compiler with these modifications has been made available to Stanford University Data Systems Laboratory for compiling a version of the internet TCP (transmission control program). This has provided a considerable saving in effort for their work and simultaneously provided valuable feedback of bugs and documentation deficiencies.

IV. COTCO ACTIVITIES

A. INTRODUCTION

During this quarter there has been continued progress in all areas covered in the plan in our original proposal. We released the initial "strawman message service test system" in April 1975 and systematically collected data on the reactions of users. A new design of the message-reading portion of the system was developed and a new design has been produced and largely implemented. A number of other features and many human factors refinements have been incorporated that were suggested by various consulting groups.

The human factors group has made a major contribution to the philosophy of the new MAILSYS message-processing design. In addition, a design study for a new Text Editor to be incorporated into MAILSYS has been almost completed.

Planning for the COTCO interactive test has continued, with consultation meetings with NAVELEX and MITRE personnel.

There has been continued progress in determining the security measures necessary to enable MAILSYS to operate in a classified environment, and in identifying and implementing the changes in the TENEX system that will be necessary to provide a secure message facility.

A statistics-gathering package which captures statistics on the MAILSYS system in use has been designed, implemented, and is now being tested.

Work is continuing on the design and implementation of the LDMX-TENEX interface protocols, in collaboration with NAVCOSSACT.

B. MAILSYS DEVELOPMENT

1. The Initial Test Period

The initial "strawman message service test system" was released to the user committee, in April 1975, as planned in the original proposal.

User reaction has been systematically collected by providing a central file to which messages of complaint and suggestions for changes could be sent.

MAILSYS makes it easy for users to send messages and the users have responded freely, sending messages that range from spur-of-the-moment problems to carefully thought-out suggestions for changes in the basic design of the system.

These messages have been collected, abstracted, and classified. A report will soon be released on the analysis process and the results of the analysis. Many of the user suggestions have already been incorporated into MAILSYS.

Several groups of users are represented in this shake-down period of general use:

(a) Navy personnel from NAVCOSSACT and NAVELEX
(b) Mitre Staff personnel
(c) BBN human factors specialists
(d) The message service community
(e) The members of the Message Group formed by Tasker at ISI.
(f) The general user population at BBN

The MAILSYS system has been continuously modified in response to user complaints and preferences throughout this period.

2. New MAILSYS Design Guided by User Reaction

In addition, as a result of studying the reactions of users to MAILSYS and other message-processing systems (particularly MSG and HG), a redesign of the message-processing part of MAILSYS has been undertaken. A number of other features and human-factors refinements have been included which were suggested by the various consulting groups.

The next version of MAILSYS will have substantially revised message-processing capabilities. It is our hope that this version will provide the user with greatly enhanced processing capabilities, while at the same time it will make simple processing tasks easy to specify and learn.

(a) Generality

The new design is structured around the recommendations of the human factors consultants that there be a small number of commands that can be used to do all message processing tasks.

These basic commands will combine with a carefully thought-out set of subcommands or arguments to give a flexible, powerful and easily understood group of functions.

In parallel with the basic commands, some of the most common patterns of use will be supported by specialized one-word commands so that the user can carry out commonly-done tasks quickly.

(b) Reduced Typing

The specialized commands will reduce the typing required to carry out tasks. Another device for reducing the amount of typing required is the concept of the "Current context" or environment, which consists of a combination of the current INBOX, current ITEM, current output conditions, and current default settings, as specified by a set of clearly defined rules. MAILSYS users will be able to leave portions of commands unspecified and allow the system to default those portions to the corresponding parts of the current environment. At any time, it will be possible for the user to display the current environment in order to check it or make changes in it

(c) Entering the MAILSYS System

After giving the command MAILSYS (CR) to the EXEC, the user will have the option of specifying which file is to be used as the INBOX.

@MAILSYS <filename> (CR)

@MAILSYS (CR) simply inputs the default INBOX MESSAGE.TXT;1.

MAILSYS then automatically prints out, on the user's terminal, a set of one-line SURVEYs of all RECENT messages -- that is, all messages that have arrived since the last time the user looked at that INBOX. This printout can be easily aborted if the user doesn't want to see it at the moment.

At the user'S option, MAILSYS will interrupt a working session with an automatic SURVEY of new incoming messages. The user will be able to set this option as part of the DEFAULT PROFILE.

MAILSYS will in the future be much more selective in accepting files as INBOXES, and will allow the system to operate only upon files that are in MAILSYS-readable form. This will overcome the difficulties encountered in the present system when users have attempted to INPUT non-message files.

(d) The Concept of the Current Item

The current item in MAILSYS is the item which is ready to be scanned and printed out, or otherwise output when the next message-processing command is given. The current item will be able to be

1. printed out "Current Item is N".
2. incremented or decremented by one.
3. entered in an item list symbolically, as "."

(e) Command Structure

In certain cases, the commands requiring several lines and the use of previously constructed FILTERS will be replaced by multipart, one-line commands. It will be possible to use "throwaway" one-time FILTERS, entered on the same line as the primary command. This will substantially reduce the burden of typing commands.

(f) Lineprinter Output

Messages printed out on the lineprinter will automatically be preceded by a list of one-line SURVEYs, forming a table of contents.

It will be possible to use MAILSYS to print out the MAILSYS on-line documentation on the lineprinter.

(g) Surveying

The new MAILSYS release will provide a better engineered survey command that will make it possible to generate selective SURVEYs showing just part of the INBOX contents.

(h) Message Classes

MAILSYS will provide a simple and uniform way of describing various classes of messages.

RECENT -- all messages that have arrived since the last time the user entered the INBOX

OLD -- messages that arrived before the user last entered the INBOX

SEEN -- messages marked SEEN

UNSEEN -- messages that have not been marked SEEN

DELETED -- messages marked for deletion

UNDELETED -- messages that have never been marked for deletion, or that have been DELETED and then UNDELETED

(i) The Status of the New Version of MAILSYS

The next version of MAILSYS has been substantially implemented, and documentation work has begun. The program and its documentation will be ready to release to the user community in the near future.

C. HUMAN FACTORS

1. Human-Factors Input to the Design of MAILSYS Commands

The human-factors group has continued consulting with the MAILSYS group on the system design, with strong emphasis on design of a clear, logical system whose basic features can be immediately appreciated by the naive user. There have been periodic meetings between the programming and human factors staffs, and each stage in the evolving system design has been submitted for review to the human factors group.

2. Design of a Text Editor

A design study for a editing system to be built into MAILSYS is nearing completion. This system will be automatically called whenever entries are being made in a message field. It will permit the user to "move around" within a message field, and to search for, replace, insert and omit text strings. Although this built-in editor will be considerably more powerful than the very limited editing capabilities available through the use of Control-A, Control-W, and Control-Q in the present MAILSYS, it will be simpler and easier to learn than the text editors, TECO and XED, which are presently available as subsystems running on a "inferior fork" in MAILSYS. For users who wish to perform complicated editing jobs, TECO and XED will remain in the system.

3. Planning for Interactive Test

The human factors group has continued planning for the interactive COTCO test along the lines described in the last progress report.

We have participated in a number of meetings with Joe Blankenheim of NAVELEX in which we reviewed system design issues.

We also met with MITRE staff members Jonathan Mitchell and Nancy Goodwin to discuss key factors in the test planning effort.

D. SYSTEM SECURITY

BBN Personnel took part in a security discussion during a two-day meeting with NAVELEX, ARPA, MITRE, ISI and NRL , in Washington, D.C., on 24-25 July 1975.

1. Identification of Security Requirements

We are working closely with MITRE staff members on the problem of security. In response to criticisms raised in the RISOS report to NBS on security aspects of various operating systems, by the ARPA Operating System Security Committee, chaired by Butler Lampson, and by various requests from the Message Technology Project, Burchfiel has prepared a memo detailing the TENEX security problems which BBN plans to pursue in preparation for the CINCPAC Interactive Test. These fall into three general classes:

(a) Correcting of bugs and weak points in the system that could expose information to unauthorized users.

(b) The setting up of a class of MAILSYS users who would have access to top-level MAILSYS only, and not to the full programming and editing capabilities of TENEX. This requires that some editing and fill-handling capabilities now handled by TENEX must be inserted into MAILSYS.

(c) Although it is not strictly a security problem, the CINCPAC test requires that MAILSYS facilities be made available to MAILSYS users who are not assigned directory space in the files, thus making it possible to service a large and shifting population of users without a correspondingly large commitment of file space.

2. Security-Related Modifications of TENEX

We have continued to implement the modifications of TENEX that are necessary to provide a secure message facility and to prepare for the CINCPAC test.

3. Security Issues and Costs of MAILSYS on MULTICS

In response to a request from ARPA-IPTO, we generated an estimate of the costs of converting and running MAILSYS on MULTICS, with attention to the cost of the MULTICS hardware, the costs of converting MAILSYS software, the necessity for MULTICS system programming to meet the requirements of the BBN and ISI message systems and the problems of security on both TENEX and MULTICS. We hope this estimate will prove helpful in ARPA's resource planning activities.

E. STATISTICS

We have designed, implemented and are completing the testing of a software package for gathering statistics on the operation of MAILSYS. This package extends the capability for gathering statistics on messages sent into and out of each TENEX host computer.

Data is collected with MAILSYS on patterns of use by the user group. Measurements of system performance have been developed, such as the time (and hence the cost) per message.

1. Nature of Statistics Gathered

Statistics are being gathered in the BBN Message System in two major areas: the first is a measure of utilization of the transmission media both with respect to the amount of traffic and the time at which time traffic occurs. The second major area is the internal instrumentation of MAILSYS, reporting the utilization of each individual command, as well as the CPU and console time required to perform major functions, such as reading and creating messages.

The statistics gathered with respect to network utilization are the following:

(a) the total number of messages received from each site during the measurement period.
(b) the total number of messages transmitted to each site.
(c) The average number of messages per day processed. This includes both incoming and outgoing.
(d) The list of users, including processes, which have sent at

least one message during the measurement period.
(e) the list of users, including processes, which have received at least one message during the measurement period.
(f) the number of messages received per half hour block per week.
(g) the number of messages sent per half hour block per week.
(h) the CPU time for the sending process averaged per message.
(i) the receiving process CPU time average per message.

The following are the internal instrumentation statistics gathered by MAILSYS itself:

(a) The average number of addressees per message
(b) The average number of different hosts that each message is being sent to.
(c) The average size of the message header in characters for outgoing messages.
(d) The average size in characters of a message for outgoing messages
(e) The percent utilization of each header item - this is defined as the percentage of the messages generated which had each individual header type field included.
(f) The count of how many times each section of Describe and/or Help has been used.

2. Data Collected as Histograms

Certain data in the MAILSYS program are being collected in the form of histograms. These are logorithic histograms with values ranging where the first bin is 0 to .2 in appropriate units, and the last bin is greater than or equal to 50,000 in the same units. The following data is being sorted into histogram bins.

(a) The number of characters per message
(b) The number of characters in each header.
(c) The number of different hosts each message is being sent to.
(d) The number of addressees per message.
(e) The console time for the creation of the message.
(f) The CPU time for the creation of a message.
(g) The console time for reading a message.
(h) The CPU time for reading a message.
(i) The CPU time for every command executed.
(j) The console time for every command executed.

For e, f, g, and h the values are figured as averages for each individual session and the value of the average determines which bin of the histogram will be incremented. The bin is incremented by the number of occurrences, either messages read or messages created, per session. We assume that over the long term this will reflect the same distribution pattern as would have occurred if the true values and not the averages were being kept.

3. MAILSYS Interface for User-Supplied Statistical Packages

Future MAILSYS users will be able to insert their own software packages into MAILSYS for the purpose of gathering statistics of special interest to them. There are three main sections in the MAILSYS system into which hooks for statistical packages have been placed: (1) at the beginning and end of the main command scanner loop to allow a user to gain timing information for commands, (2) in the SEND command code execution routines, which give information as to the message having been sent, and (3) in the Describe package.

F. LDMX INTERFACE

We have collaborated with NAVCOSSACT to design the details of protocols of the proposed LDMX/TENEX interface.

1. Overall control of LDMX/TENEX traffic is in the hands of a logged-in Operator job, with controlling terminal likely at the LDMX site. This job has absolute socket capability.

The Operator Job:

(a) enables the operator to set up and initialize the LDMX/TENEX channel.
(b) processes both directions of byte traffic between LDMX and TENEX, and supports lower levels of RIXT block protocol.
(c) reassembles incoming blocks from the LDMX into full messages and sends them to the TRANSLATOR (an autostart TENEX job) .
(d) disassembles outgoing messages from the TRANSLATOR into blocks and sends them to the LDMX.

The TRANSLATOR job is to translate from ACP126 to and from RFC680 messages. The TRANSLATOR connects to a COTCO TENEX users job which contains COTCO MAILSYS (or other message process) that creates and/or processes messages in the usual way.

2. A key issue is translation between TENEX and LDMX addresses, and the handling in TENEX of messages addressed to LDMX users. "ADDRESS XLATION TABLE" provides the data base for this purpose. This table is maintained by LDMX/TENEX operator, and is made available to COTCO MAILSYS for create-time address checking.

The TRANSLATOR is also made available to COTCO MAILSYS for general prescan of message format.

Four types of error processing are provided.

3. Network Configuration: LDMX traffic can be isolated from general ARPANET traffic, or the LDMX interface can be treated as a source and receiver of messages equivalent to any other ARPANET user.

4. The TENEX portion of these protocols has been 35 per cent designed, 25 per cent implemented.

G. DISTRIBUTION SYSTEM: DESIGN OF MESSAGE TRANSMISSION PROTOCOL

We have collaborated with the Message Service Committee in designing a tentative Message Transmission Protocol (MTP) for structured messages using the PCPB8.

We have prepared a critique of the the initial MTP and have made an alternative proposal for an MTP which provides system support at a more primitive level, but allows for correspondingly greater freedom and power in the upper applications levels.

1. The Mandate of the Message Service Committee

The mandate of the MTP SubCommittee of the MSC, as BBN understands it, was to propose a new protocol for sending structured messages in the network. We interpret this mandate to imply a request for two things: (1) a definition of, and an encoding for, "structured messages", and (2) a definition of, and protocol for, sending messages (of all kinds) in the network. Also implied in the mandate is a request for a specification for representing RFC680 messages as "structured messages".

2. BBN Objections to the Proposed MTP Protocol

This section summarizes BBN´s current position on the proposed new Message Transmission Protocol (MTP) (Haverty, July 8, 1975).

(a) It does not respond to the mandate of the Message Service Committee (MSC) in that it does not support the transmission of arbitrary structured objects.

(b) It addresses a collection of problems not fundamental to the future development of message-based communication in the network. In doing so, MTP overlooks a simple solution to the problem of encoding structured objects, adopting the overly-complex view that messages are changing objects which may appear differently to different receivers of them.

(c) It leaves untouched the critical issue of extending the currently-primitive notion of message transmission services to include process-to-process service in addition to the user-to-user service.

3. Outline of BBN's Alternative Proposal

An alternative and simpler response to the MSC's mandate is as follows:

(a) Regard messages as unchanging objects. Then the servers which transmit messages will not need to understand their structure. This permits separating the structure of messages from the transmission of messages.

(b) To support the creation of server processes which communicate quickly by sending messages to one another (as exemplified by the MTP protocol), build a high-speed message transmission system (MTS) whose function it is to transmit messages which it views simply as collections of bits.

(c) Adopt PCPB8 as an encoding for structured objects when structured objects are to be transmitted. This defines structured objects as those objects which can be encoded in PDPB8.

A message transmission service (MTS) is responsible for delivering a message (any bit string) to a collection of receivers. To give this meaning, we must say who receivers are. "User at host" is an inconvenient way to name processes. Also it ties a user or a process to a particular host. We, therefore, feel that this is a good time to broaden our view of message transmission by changing the notion of "address". A new message transmission service (MTS) should be provided in the network by having co-operating servers in addition to managing network connections over which messages are transmitted, maintain a network-wide collection of addresses. A process could request an address which it could supply to other processes to have them send messages to it. By making addresses be 72 (or 144) bits long, there should be no need to re-use addresses for any reasonable duration of the MTP system (say 100 years).

A number of extensions of this MTP service are imaginable. Associated with certain addresses could be character strings (names of people, subsystems, services). The MTP service could be queried to obtain addresses whose associated strings meet certain criteria. Also associated with an address could be a program to be run when a message arrives for that address. There need be no requirement that addresses be host-dependent; that is, the service could be extended to be able to move an address to another host. Conceivably, addresses which had moved might receive their messages a bit slower, due to forwarding. The service can be designed to answer a reasonable set of queries concerning addresses (location, strings, programs), speed of service (hot temporarily down), cost of service, and the like.

Since the MTS serves of [illegible] and low-speed communication, there is a need for some method of guaranteeing that high-speed messages not be held up by slow-speed ones. That is, the MTS should understand message priority. What is needed then, is a protocol